Anyfeeder Integration Sample

YASKAWA

Table Of Contents

Revision History

Date	Version	Author	Modification
01.10.2018	1.00	Trapp	Initial version

Table 1 Revision History

1 Introduction

The Anyfeeder integration is based on Yaskawa's MotoGSI and handles the communication to the Anyfeeder system. It's provided as a sample application. Thus Yaskawa can't provide free of charge support in case of problems. The sample code was running on a DX200 so it might be that the jobs can't be loaded on others platforms without smaller changes due to syntax incompatibilities. Since most of the syntax is compatible the changes should not be very time consuming. MotoGSI is freely available as package for different controller platforms starting from DX100.

2 Controller Requirements

- MotoPlus runtime (this is a paid option on some controllers)
- MotoGSI (free available)
- Working ethernet connection to the Anyfeeder

3 Overview

The sample jobs are based on the Anyfeeder scheme described in Cognex-Robot Communication Rev I_de.pdf

The sample contains a set of jobs which are structured as following

3.1 GSI_....JBI jobs

These jobs are the basic GSI jobs from the GSI library required to run the sample. These jobs must not be changed or edited by the user

3.2 ANYFEEDER_.....JBI

These jobs are the basic Anyfeeder communication jobs based on GSI. These jobs must not be changed or edited by the user

3.3 G_FIND_PART.JBI and G_KALIBRIERUNG.JBI

These jobs need to be adapted by the user to fit their application.

4 Main user jobs

4.1 G_KALIBRIERUNG.JBI

This job calibrates the robot to the Anyfeeder device. The robot positions must be adapted.
Please check the comments inside the job.

4.1.1 Remarks to comments used in the job

*1	In the sample code OT#3 is used to control the gripper.
*2	Starting position of the robot.
*3	Connect the Anyfeeder Device. Please set the right IP address.
*4	Section to place back the part from the image section to the cluster. In the sample job this was done by hand so there is no robot movement included. To fully automate place the code to pick the part and place it back to the cluster here
*5	This is basically the start of the calibration. The part must be placed at 4 different points inside the camera section with 4 different angles (0,90,180,270 deg). The positions must be adapted. Basically the sequence is always the same • Pick the part from the cluster • Place it on Point I with Angle W • Store the robot position • Move the robot out of the image section The robot position in the sample is stored based on User Frame 1 which defines the anyfeeder work area.
*6	Stores the calibration data. Use your calibration job data.
*7	Disconnect the Anyfeeder Device

4.2 G_FIND_PART.JBI

This job reads out the found parts from the Anyfeeder device and picks the part. The robot positions must be adapted. Please check the comments inside the job.

4.2.1 Remarks to comments used in the job

*1	In the sample code OT#3 is used to control the gripper
*2	Starting position of the robot.
*3	OT#18 is connected to the feeder lock signal. Check the feeder documentation
*4	Connect the Anyfeeder Device. Please set the right IP address
*5	Load the feeder job. Use your feeder job.
*6	Get the position of the found parts from the feeder. The feeder supports also a multi pick mode. In the sample only a single position is read. The method provides an error in case no part is found for a longer time period.
*7	Turn on the feeder lock to move the robot inside the camera section.
*8	Create a P variable with the current data of the found part. The Anyfeeder returns X,Y, and an angle around Z
*9	Move the robot to the found position. Pick the part and place it where required.
*10	Move the robot back to the start postion.
*11	Turn of the feeder lock when the robot is outside the feeder section
*12	Close the connection

5 Anyfeeder Communication Library

5.1 Requirements

The following variables can‘t be used because they are used for internal tasks

- BVars from B100 to 300
- IVars from B100 to 200
- SVar from 0 to 10

5.2 Functions

5.2.1 ANYFEEDER_CONNECT “IN_IPADDRESS”

Connection to Anyfeeder

RET:
>= 0 Handle id (The ID has to be used for communication functions)
-1 Error connection

5.2.2 ANYFEEDER_DISCONNECT “IN_HANDLE”

Disconnect Anyfeeder device

5.2.3 ANYFEEDER_FORMAT_CALIBDATA “IN_STRING” “OUT_IDX_DVAR”

Formats the calibration data

IN_STRING Input calibration data string (String returned by GVL000)
OUT_IDX_DVAR Index of DVar which holds the result data if RET=2

RET:
-1 No data found
0 CalibZur
1 NoPart
2 CalibDataFound (if String is e.g. Calib3;C270.000 the D[OUT_IDX_DVAR] holds 3 and D[OUT_IDX_DVAR+1] holds 270)

5.2.4 ANYFEEDER_GETPICKRESULT “IN_HANDLE” “OUT_IDX_VAR”

Readout multi pick result.

IN_HANDLE	Connection handle
OUT_IDX_VAR	DVar index to write the result data
D[OUT_IDX_VAR]	Found Parts (max 2)
D[OUT_IDX_VAR+1]	X Part1
D[OUT_IDX_VAR+2]	Y Part1
D[OUT_IDX_VAR+3]	C Part1
D[OUT_IDX_VAR+4]	X Part2
D[OUT_IDX_VAR+5]	Y Part2
D[OUT_IDX_VAR+6]	CPart2

RET:

1	No Part
0	Part found

5.2.5 ANYFEEDER_GETVALUE “IN_HANDLE” “IN_TYPE” ”IN_COMMAND” “OUT_IDX_RESULT”

IN_HANDLE	Connection handle
IN_TYPE	0 = int 1=float 2=String
IN_COMMAND	cell to readout (e.g. h003)
OUT_IDX_RESULT	Index to write result based on IN_TYPE

RET:

0	OK
-1	Send failure
-2	Read cell failure
-3	Timeout
-4	Cell is empty

5.2.6 ANYFEEDER_SETPARAM “IN_HANDLE” “IN_COMMAND”

Write value to Anyfeeder

IN_HANDLE	Connection handle
IN_COMMAND	Command to set (e.g. SO1)

RET:

0	OK
-1	Not OK
-2	Send error
-3	Error get result

5.2.7 ANYFEEDER_CONVERT_FLOAT “IN_STR_COM” “IN_DVAR” “OUT_IDX_STR”

Convert the input DVar to a command with float value

IN_STR_COM	(e.g. SFP000)
IN_DVAR	(e.g. 300123)
OUT_IDX_STR	S[OUT_IDX_STR] holds result (e.g. SFP000300.123)

Imprint

Yaskawa Europe GmbH
Yaskawastraße 1
85391 Allershausen
Germany
Phone 00498166900
Fax 0049816690103